

Plugin Languages

Using Weather_routing_pi as an example

Notes from Crowdin Source Maintainers:

Pavel [nohal] “owns” OpenCPN Crowdin and both he and Dave [bcdcat] will of course maintain Crowdin. [Hakan] offers his help to relieve the pressure on Pavel and Dave. So he'll of course do what he can whenever possible and needed.

What a plugin developer has to do is give us a message whenever the code is ready enough to update Crowdin, that's the main issue. If potfiles.in and the xxx_pi.pot is also updated in source it's very fine. But the message to any or all of is the main task.

Then we will try to watch Crowdin for updated translations but a forum note from translators is of course also appreciated.

Communications with Source Maintainers:

1. Plugin developer determines when to advise Hakan, Pavel & Dave when there are going to be no more new strings so that a *.POT file can be developed.
2. Hakan, Pavel or Dave develop the [plugin].pot file and post it to Crowdin announcing a deadline by which language updates must be ready.
3. Hakan, Pavel or Dave make a Pull Request to the plugin developer with the completed language *.PO files.
4. Plugin developer merges the Pull Request and then the languages become available when the plugin in is built the next time.

Crowdin Project OpenCPN - Translators are needed

<https://crowdin.com/project/opencpn>

Translators are needed. Please register and login, to help us translate.

[We need a description of how this is done in Crowdin in this section.]

1. Once you have logged into the Crowdin OpenCPN Project, you can select a language.
2. You will be presented with an outline of OpenCPN and each of the plugins that are available.
3. These sections show the % Completed. Clicking on a section, will allow you to add translations word by word or phrase by phrase.
4. When the translations have been completed for OpenCPN or one of the plugins, the developers then move the completed files into the “build” process to embed the translations into OpenCPN or the Plugins.

This is not an instantaneous thing and requires communication and coordination with the translators. So translators should leave messages in the forum when they are done. We suggest using the [Internationalization thread](#) for leaving messages of this nature.

Basic Steps Required to create Language files

as described by Gerhard.

Create or Update the POTFILES.IN file

The first job is to make or update the POTFILES.IN which is simply a list of all source file names that have language strings. This file lives in the PO directory.

There appears to be no real difference between Windows and other operating systems for making po/mo files. You need the “gettext” command, make a search for gettext for Windows in Google. Also you need to know how the Terminal works, especially the path handling. The Compiling thread has instructions for installing gettext.

Then first make an ASCII file called POTFILES.in when not available already, which holds all source file names of your plugin line by line, but only those files which may contain language strings. These language strings are preceded with `_("` , that is underscore, bracket open and quotation mark. e.g.: `_("This is a text for translations.")`

In the Terminal you use the gettext command like this:

```
gettext -from-code=iso-8859-1 -force-po -package-name=xyz_pi -package-version=1.0 -output=xyz_pi.pot -keyword=_ -width=80 -files-from=POTFILES.in
```

So the parameters for gettext are

- from-code= the kind of code
- force-po to force a po file
- package-name= the name of the plugin
- package-version= the version of the plugin
- output= the name of the resulting pot file
- keyword=_ as explained above
- with=80 the length
- files-from= POTFILES.in the list of files to be processed

If you think there are strings missing in the resulting pot file go to the plugin source files and look there for these strings. Perhaps they are preceded with `_T("` or `wxT("` and not with `_("`

Updating an Old PO File

If you have already an old po file you can merge it with your new pot file using the msgmerge command which is part of the gettext bundle. Use msgmerge -help in the Terminal window to see how it works.

Making an new PO File

Otherwise simply make a copy of your new pot file with the extension po to receive an empty po file for the translation work with Poedit.

Preparing for Crowdin Source Language File updates for Plugins

as described by Hakan

1. Update the "**potfiles.in**". See Gerhard's description above.
2. Then produce a fresh "**weather_routing_pi.pot**" file.
3. With a new updated pot-file Hakan can update Crowdin in a minute, so that translators can complete the work.

So unless no one can update potfiles.in and produce a fresh "weather_routing_pi.pot" file

Use VS2013 to "Build only" the POT project for the Plugin

In VS2013 use "Build only" of the POT file project, but first be sure that potfiles.in is updated. Hakan needs the fresh pot file to update Crowd Source.

1. Open Visual Studio 2013
2. Open weather_routing_pi.sln located in the build directory of weather_routing_pi local git directory.
3. Find in Solution Explorer "weather_routing-pot-update" and expand it there should be cmake rules, etc.
4. Right click on "weather_routing-pot-update" and select the third down "Project only", then select "Build Only "weather_routing-pot-update".
5. Output window shows Build: 1 Succeeded 0 Failed
(Did the same with "weather_routing-po-update" which builds only the po files.)

You may Upload the [Plugin_pi].pot file to Hakan via the forum as attachment or via email. weather_routing_pi.pot (take off the pdf)

So I just uploaded the file to the forum thread [here as an example](#) and very soon afterwards Hakan reported that the files on Crowd Source were updated and ready for the Translators.

Next step will be to collect the *.PO files and compile them with the plugin.

Plugin usage of own catalog for Language "strings"

wxWidgets provide a macro to help with the translation of strings into the local language. This is done using the macro, normally seen as `_("string")` in the code. At compile time this is translated to `wxTranslation("string")` which at runtime will return the string "string" translated into the local language. The translation is done by matching the string "string" in the language catalog set (all the catalogs that have been loaded for OCPN). The first match that is found is returned.

It is possible for each plugin to use the same words for different meanings, eg. "Port" can mean "Communications Port", "Harbor", "Port of Entry", "Port Side", "Porthole", etc. Unfortunately as the first instance of the word is returned the translation may not be appropriate. The upside of this is if you use strings that have not been translated in your catalog, then it may have been in another and this translation will be returned. The downside is that you cannot specify, using wxWidgets macro, which catalog to use.

The wxTranslation method comes in two flavors `wxTranslation("string")` `wxTranslation("string", "catalog")` You could use this in your code, but it would be long winded. It is possible to redefine the macro such that it will use the second form if "catalog" is specified. The following shows how this can be done to allow the code to maintain the look of all the other code in OCPN and the Plugins.

Changes to CMake file

If you use Pavel's/Sean's Plugin cmake file set you will have a file called PluginConfigure.cmake in the "cmake" directory of your project. This file needs a new line adding:

```
configure_file(cmake/wxWTranslateCatalog.h.in
${PROJECT_SOURCE_DIR}/include/wxWTranslateCatalog.h)
```

This line is added after the line:

```
configure_file(cmake/version.h.in ${PROJECT_SOURCE_DIR}/include/version.h)
```

New template file

The name of the input template file and the output header file can be changed. The name was picked to signify "wxWidgets Translate with Catalog".

The content of the input template file is:

```
*****
* $Id: wxWtranslateCatalog.h,v 1.0 2015/01/28 01:54:37 Jon Gough Exp $
*
* Project:   OpenCPN
* Purpose:   Redefine _() macro to allow usage of catalog
* Author:    Jon Gough
```

```

*
*****
*   Copyright (C) 2010 by David S. Register   *
*   $EMAIL$                                   *
*
*   This program is free software; you can redistribute it and/or modify *
*   it under the terms of the GNU General Public License as published by *
*   the Free Software Foundation; either version 2 of the License, or *
*   (at your option) any later version. *
*
*   This program is distributed in the hope that it will be useful, *
*   but WITHOUT ANY WARRANTY; without even the implied warranty of *
*   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the *
*   GNU General Public License for more details. *
*
*   You should have received a copy of the GNU General Public License *
*   along with this program; if not, write to the *
*   Free Software Foundation, Inc., *
*   51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA. *
*****
*/
#ifndef _ODCATTRANS_H_
#define _ODCATTRANS_H_

#ifndef WXINTL_NO_GETTEXT_MACRO
#ifdef _
#undef _
#endif // _
#if wxCHECK_VERSION(3,0,0)
#define _(s) wxGetTranslation((s), wxS(__GESHI_QUOT__opencpn-
${PROJECT_NAME}__GESHI_QUOT__))
#else // wxCHECK_VERSION(3,0,0)
#define _(s) wxGetTranslation(wxT(s), wxT("opencpn-${PROJECT_NAME}"))
#endif // wxCHECK_VERSION(3,0,0)
#endif // WXINTL_NO_GETTEXT_MACRO

#endif
''

```

This will generate a header file in the include directory that has the following content:

```

''/*****
***
* $Id: wxWTranslateCatalog.h,v 1.0 2015/01/28 01:54:37 Jon Gough Exp $
*
* Project: OpenCPN
* Purpose: Redefine _() macro to allow usage of catalog

```

```
* Author:   Jon Gough
*
*****
*   Copyright (C) 2010 by David S. Register   *
*   $EMAIL$                                   *
*
*   This program is free software; you can redistribute it and/or modify *
*   it under the terms of the GNU General Public License as published by *
*   the Free Software Foundation; either version 2 of the License, or   *
*   (at your option) any later version.                                         *
*
*   This program is distributed in the hope that it will be useful,         *
*   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
*   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
*   GNU General Public License for more details.                             *
*
*   You should have received a copy of the GNU General Public License      *
*   along with this program; if not, write to the                          *
*   Free Software Foundation, Inc.,                                         *
*   51 Franklin Street, Fifth Floor, Boston, MA 02110-1301,  USA.          *
*****
*/
#ifndef _ODCATTRANS_H_
#define _ODCATTRANS_H_

#ifndef WXINTL_NO_GETTEXT_MACRO
#ifdef _
#undef _
#endif // _
#define _(s) wxGetTranslation((s), wxS(__GESHI_QUOT__opencpn-
your_plugin_pi__GESHI_QUOT__))
#else // wxCHECK_VERSION(3,0,0)
#define _(s) wxGetTranslation(wxT(s), wxT("opencpn-your_plugin_pi"))
#endif // wxCHECK_VERSION(3,0,0)
#endif // WXINTL_NO_GETTEXT_MACRO

#endif
'
```

This header needs to be included in every source module that is going to do translations. It should be placed after the wxWidgets includes to ensure that it can redefine the macro.

Changed template file

The “version.h.in” used by this cmake process also needs to be updated with two new lines:

```
#define PLUGIN_NAME ${PROJECT_NAME}
```

```
#define PLUGIN_CATALOG_NAME wxS("opencpn-${PROJECT_NAME}")
```

The whole "version.h.in" file should look like:

```
#define PLUGIN_NAME ${PROJECT_NAME}

#define PLUGIN_CATALOG_NAME wxS("opencpn-${PROJECT_NAME}")

#define PLUGIN_VERSION_MAJOR ${VERSION_MAJOR}

#define PLUGIN_VERSION_MINOR ${VERSION_MINOR}

#define PLUGIN_VERSION_PATCH ${VERSION_PATCH}

#define PLUGIN_VERSION_DATE "{VERSION_DATE}"
```

Change to main module

To implement this the code you use to set the locale catalog needs to change from:

```
AddLocaleCatalog( opencpn-your_plugin-name_pi );
```

to:

```
AddLocaleCatalog( PLUGIN_CATALOG_NAME );
```

The "PLUGIN_CATALOG_NAME" is translated at compile time to the correct name for your plugin. This ensures that you are using the same catalog as that added to OpenCPN. This is really a failsafe.

How to make wxFormBuilder use new wxWTranslateCatalog.h file

This is quite simple. Once you have created the wxWTranslateCatalog.h file you can include it in all generated header files by a single change to the project file.

Open wxFormBuilder with the project file you want to change, then go to Properties/C++ Properties/class_decoration/header and enter wxWTranslateCatalog.h in that field. This will append this header file after all the wxWidgets headers and allow it to redefine the "_"() macro to use the project local catalog. Now there is no need to change the generated file to make it work.

Analysis and Avoidance of Duplicate Language "strings" (words or phrases)

Proposed by NAV.

More important is probably the simple procedure I used to make it (ready in a few minutes):

1. Go to GITHUB;
2. Find the correct repository (master branch);
3. Open po-file of the language you're interested in;
4. Click on "Raw";
5. Copy all;
6. Paste to two different sheets (e.g. 1 and 2) of Excel;
7. Delete all empty rows on sheet 1 (use Excel Add On to do this with a few mouse clicks);
8. Filter on "msgid" in sheet 1;
9. Delete all hidden rows on sheet 1 (use Excel Add On to do this with a few mouse clicks);
10. Delete all empty rows on sheet 2;
11. Filter on "msgstr" in sheet 2;
12. Delete all hidden rows on sheet 2;
13. Copy all msgstr rows from sheet 2 to next empty column in sheet 1;
14. Check results.
15. Add plugin name in separate column for all copied rows;
16. Add sequential number in separate column to be able to re-sort.

The Excel Add On I use adds additional functionality, of which I only used two for this procedure. You can find it here: [ASAP Utilities](#)

From:
<https://opencpn.org/wiki/dokuwiki/> - **OpenCPN Manuals**

Permanent link:
https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:developer_manual:developer_guide:internationalization:plugin_languages

Last update: **2017/01/06 19:18**

